LiveCode Builder Bytecode Reference

Introduction

The LiveCode Builder Virtual Machine is a infinite register machine with a high-level and verifiable bytecode.

All bytecode runs in the context of a module instance with executing handlers forming a stack of activation frames. Each frame contains an array of registers, the first part of which are parameters followed by handler-local variables and bytecode block registers.

Most bytecode operations operate directly on registers, access to module level definitions (handlers, constants, variables) are indirected through the fetch and store operations.

Each bytecode operation has an address which can be jumped to using the jump operations.

Operations

Jump

```
jump <label>
```

The jump operation sets the address of the next instruction to execute to that identified by **.**

Jump If False

```
jump_if_false <register>, <label>
```

The jump_if_true operation checks the value in and jumps to **if it is 'false'.**

If is a runtime error if does not contain a boolean.

Jump If True

```
jump_if_true <register>, <label>
```

The jump_if_true operation checks the value in and jumps to **if it is 'true'.**

If is a runtime error if does not contain a boolean.

Assign Constant

```
assign_constant <register>, <constant>
```

The assign_constant operation copies into . Here can be the nothing literal, the true or false literal, an integer literal, a real literal, a string literal, a list literal or array literal.

If is typed then it is a runtime error if the type of does not conform to that type.

Assign

```
assign <dst-register>, <src-register>
```

The assign operation copies the value in to .

If is typed then it is runtime error if the type of the value in does not conform to that type.

Return

```
return [ <result-reg> ]
```

The return operation exits the current handler, returning to the instruction after the invoke operation which created it. If present, the value in is copied as the return value. If not present then the return value will be 'nothing'. Additionally any values in out or inout parameters are copied back to the caller.

If is not present, then the return value will be 'nothing'.

It is a runtime error if the type of the return value does not conform to the return type of the current handler's signature.

It is a runtime error if any out or inout parameters are unassigned at the point of return.

Invoke

```
invoke <handler>, <result-reg>, <arg1-reg>, ..., <argn-reg>
```

The invoke operation creates a new frame copying values from the argument registers for any in or inout parameters. It then starts executing the bytecode attached to , a definition. The return value is placed into the register .

If is a runtime error if the number of arguments provided is different from the signature of

If it a runtime error if for in and inout parameters, the contents of does not conform to the type of the parameter required by the signature.

Invoke Indirect

```
invoke <handler-reg>, <result-reg>, <arg1-reg>, ..., <argn-reg>
```

The invoke indirect operation functions identically to the invoke operation except that it calls the handler in .

It is a runtime error if does not contain a handler value.

Fetch

```
fetch <dst-register>, <definition>
```

The fetch operation copies the value of into . The

may be a variable, constant or handler. In the case of a handler, a handler value is created.

It is a runtime error if the type of the value of does not conform to the type of .

Store

```
store <src-register>, <definition>
```

The store operation copies the contents of into . The

must be a module level variable.

It is a runtime error if the type of the value in does not conform to the type of .

Assign List

```
assign_list <dst-reg>, <element1-reg>, ..., <elementn-reg>
```

The assign_list operation builds a list value from up to

.

It is a runtime error if the type of does not conform to list.

Assign Array

```
assign_array <dst-reg>, <key1-reg>, <value1-reg>, ..., <keyn-reg>, <valuen-reg>
```

The assign_array operation builds an array value from each key value pair

, up to , .

It is a runtime error if the type of does not conform to array.

Reset

```
reset <reg>
```

The reset operation performs default initialization of . If the type of

has no default value, it reverts to unassigned.